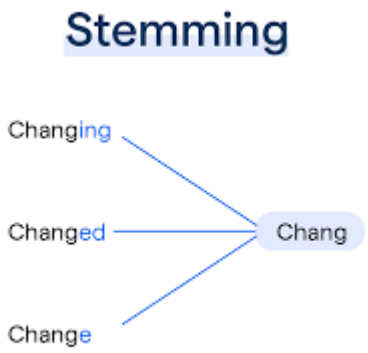
# 5. Stemming

One of preprocessing steps in Natural Language Processing(NLP) used to reduce words to their base/root form.
This helps to improve the performance of NLP model by reducing the complexity and size of the dataset.




## Overview

Process of reducing the word into its root form by stripping suffixes or prefixes.
This involves a **crude heuristic process** that chops off the ends of words in the hope of achieving the goal correctly most of the time.
**However, stemming might not always produce actual words.**

> **Crude Heuristic Process**
>
> A crude heuristic process is a pragmatic approach to problem solving that uses a mental shortcut or rule of thumb. It's not fully optimized or rationalized, but it's considered "good enough" as an approximation.
> Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of **achieving this goal correctly most of the time**, and often includes the removal of derivational affixes.

```
Derivational Affix : A morpheme that changes the meaning of a word by attaching to it to form a new word. They change the meaning of words, such as changing adjectives into verbs, nouns into verbs, or verbs into adverbs. (Ex: "-ation", "un-", "anti-", "pre-")
```

Ex:
Running -> Run

Cats -> Cat
Better -> Bett (Works most of the time, can't guarantee all the time)

**Popular Stemming Algorithms,**

1. Porter Stemmer
2. Snowball Stemmer
3. Lancaster Stemmer

## 1. Porter Stemmer

Developed by Martin Porter in 1980.
One of the most widely used algorithms.
Uses a series of rules to iteratively strip suffixes from words, transforming them into their root forms.

*Benefits,*

- Well tested and integrated into many NLP libraries.
- Offers a good balance between speed and accuracy.

*Drawbacks,*

- Over-stemming
  Sometimes removes too many characters, which can lead to non-root words.
- Under-stemming
  May not reduce some words to the same root.

*Use cases,*
Good for General Test Processing tasks.

| Word | Stemmed output |
|---|---|
| caresses | caress |
| ponies | poni |
| ties | ti |

## 2. Snowball Stemmer(Porter2)

An improved version of the Porter Stemmer.
Developed by the Martin Porter(Developer of Porter algo) and offers more accurate stemming with a more extensive set of rules.

*Benefits,*

- More accurate than the original porter algorithm.
- Retains efficient while improving the rule set.

*Drawbacks,*

- More complex than the original porter algo.
  Might slightly impact the performance.

*Use cases,*
Suitable for applications requiring more accurate linguistic preprocessing.
Ex: Linguistic Researches

| Word | Stemmed output |
|---|---|
| caresses | caress |
| ponies | poni |
| ties | tie |

## 3. Lancaster Stemmer(AKA 'Paice-Husk Stemmer')

A more aggressive Stemming algorithm.
Uses an iterative approach with a large set of rules and allows for the specification of custom rules.

*Benefits,*

- Highly aggressive in reducing words to their root form.
- Allow for the custom rule definitions.

*Drawbacks,*

- Over-Stemming
  More likely to over-stem, resulting in non-root or overly reduced words.
- Less common in general usage.

*Use cases,*

- Suitable for Highly Redundant Data(Useful in scenarios where reducing word variations aggressively is beneficial)
  Ex: some information retrieval tasks.

| Word | Stemmed output |
|---|---|
| caresses | caress |
| ponies | poni |

| Word | Stemmed output |
|---|---|
| ties | tie |

---

## Summary

| Feature | Porter Stemmer | Snowball Stemmer | Lancaster Stemmer |
|---|---|---|---|
| Accuracy | Moderate | High | Low to moderate |
| Aggressiveness | Moderate | Moderate | High |
| Performance | Fast | Fast | Fast |
| Complexity | Simple | Moderate | Simple to moderate |
| Customizability | Low | Low | High |

For general purposes, the Porter Stemmer provides a good balance of speed and accuracy. The Snowball Stemmer offers better accuracy and is suitable for tasks where more precise stemming is necessary.
The Lancaster Stemmer is best used when aggressive stemming is required, but care must be taken to handle potential over-stemming issues.

**Better to use Porter Stemmer for the Project.**

```
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def stem_text(words): #words is a list of words
        return [stemmer.stem(word) for word in words]

words = lemmatize_text(words)
```